Canazei 2018
Winter School on Inequality and Social Welfare Theory

R TUTORIAL

paolo.brunori@unifi.it

Model selection

2.1 Validation set

We may be interested in decomposing inequality by sources or by groups. An example of this the estimation of inequality of opportunity. In its simplest version this estimation consists in the measurement of inequality in the predicted values obtained by a linear regression. The dependent variable is typically income and the independent variables are circumstances beyond individual control such as area of birth or socioeconomic background.

In the 1993 Italian dataset we have, together with income, a number of exogenous determinants of income: father and mother education, father and mother occupation and area of birth. So inequality of opportunity estimate can be obtained as:

```
data<-read.dta("[your directory]Brunori_IT93.dta")

# inequality of opportunity
mod<-lm(y~m_edu+f_edu+f_occ+m_occ+birth_area, data=data)
summary(mod)
yhat<-predict(mod, data=data)
IOp<-ineq(yhat,0, type="entropy")/ineq(data$y,0, type="entropy")
IOp
```

Now, in the dataset we have two possible ways of coding the variable “area of birth”: 20 regions and 5 macro-area, what happens if we use the latter?

```
# alternative model

mod.area<-lm(y~m_edu+f_edu+f_occ+m_occ+birth_area2, data=data)
summary(mod.area)
yhat.area<-predict(mod.area, data=data)
IOp.area<-mean(log(mean(yhat.area)/yhat.area))
IOp.area
```

As expected we get a different result. Which one is to be preferred? A possible criterion to decide is to compare the accuracy of the two models using a validation set. We choose the model that minimizes the mean squared error when the models is used to predict the dependent variable out-of-sample. To do so the original sample is divided in two subsamples: a training sample and a test sample. The two models are estimated on the training sample. The parameters of both models are stored and used to predict the dependent variable in the test sample. The mean squared error (MSE) in the test sample is an unbiased estimate of the MSE out-of-sample (see Gareth et al. (2013) Chapter 5 for an introduction to model validation.)

We are going to implement it in R.

```
set.seed(1)

tr<-sample(nrow(data), nrow(data)/2)
te<-(1:nrow(data))[-tr]

train<-data[tr,]
test<-data[te,]

mod<-lm(y~m_edu+f_edu+f_occ+m_occ+birth_area,  data=train)
MSE_region<-mean((test$y-predict(mod, data=test))^2)

mod.area<-lm(y~m_edu+f_edu+f_occ+m_occ+birth_area2, data=train)
MSE_area<-mean((test$y-predict(mod.area, subset=test))^2)

if (MSE_region<MSE_area){
 print("chose region")
 } else {
 print("go for macro-area")
}

MSE_area/MSE_region
```

2.2 Cross validation

One of the weaknesses of the validation-set approach is that its results can be sensitive to the random assignment of observations to the test or training sample. Moreover, only half of the observation are used to estimate the model. This can induce an overestimation of the out-of-sample MSE, penalizing models with more regressors in favour of more parsimonious models.
A more popular approach to estimate out-of-sample MSE is cross validation. *k*-fold cross validation is based on partition of the original data on a number of non-overlapping subsamples (generally *k* ranges between 2 and 10). The algorithm produces *k* estimates of the MSE for each model. Each estimate is obtained removing the *k-th* fold (the test sample), estimating the model on all the remaining observations, and then predicting the dependent variable on the test sample. Out of sample MSE is estimated averaging the *k* MSE estimates. The package "boot" contains a function to implement *k*-fold cross validation for GLM models (which is equivalent to an OLS when the option "family" is not specified).

Setting *k* equal to the number of observation we get the so called "Leave-One-Out Cross-Validation" (LOOCV).

```
library(boot)

glm.fit=glm(y~m_edu+f_edu+f_occ+m_occ+birth_area,data=data)
MSE_region <-cv.glm(data, glm.fit)$delta[1]
glm.fit.area=glm(y~m_edu+f_edu+f_occ+m_occ+birth_area2,data=data)
MSE_area <-cv.glm(data, glm.fit.area)$delta[1]

MSE_area/MSE_region
```

```
Setting k=n may be computationally infeasible. Moreover, LOOCV tends
to underestimate the MSE because the observation used in the test
sample are almost identical each time.

# K-fold Cross-Validation

k<-10
set.seed(111)
glm.fit=glm(y~m_edu+f_edu+f_occ+m_occ+birth_area,data=data)
MSE_region <-cv.glm(data,glm.fit,K=k)$delta[1]

glm.fit.area=glm(y~m_edu+f_edu+f_occ+m_occ+birth_area2,data=data)
MSE_area <-cv.glm(data,glm.fit.area,K=k)$delta[1]

MSE_area/MSE_region


# try with k=3, k=5 and k=10
```

2.3 Best subset selection

The number of models that one can specify explodes with the number of potential regressors and number of possible way of coding the same variable. Statistical learning algorithms provide ways to search for the best possible model without having to cross-validate all possible models. We consider here one example for linear models called best subset selection. There are many alternatives to this approach you can read though Garet et al. (2013) to understand whether one or more machine learning algorithm can solve your problem.

Best subset selection is nothing else than a trick to reduce the number of models for which one has to calculate K times the MSE. Assume we have p potential regressors and we restrict our analysis to linear models (no interactions). To cross validate all possible models means to estimate $K \times 2^p$ MSE. For 10 folds and 10 regressors the number of MSE to be estimated is above 10,000. This number is reduced to 1,100 if one follows the best subset selection algorithm:

1. For all $j = 1, \dots, p$:
2. Fit all $\binom{p}{j}$ models that contains exactly j predictors
3. Call the model with highest R^2 the "*j-th* best" model
4. Select by cross validation the model that minimizes MSE among "best" models

The trick works because for given number of regressors the R^2 is a good criterion to judge out-of-sample MSE.

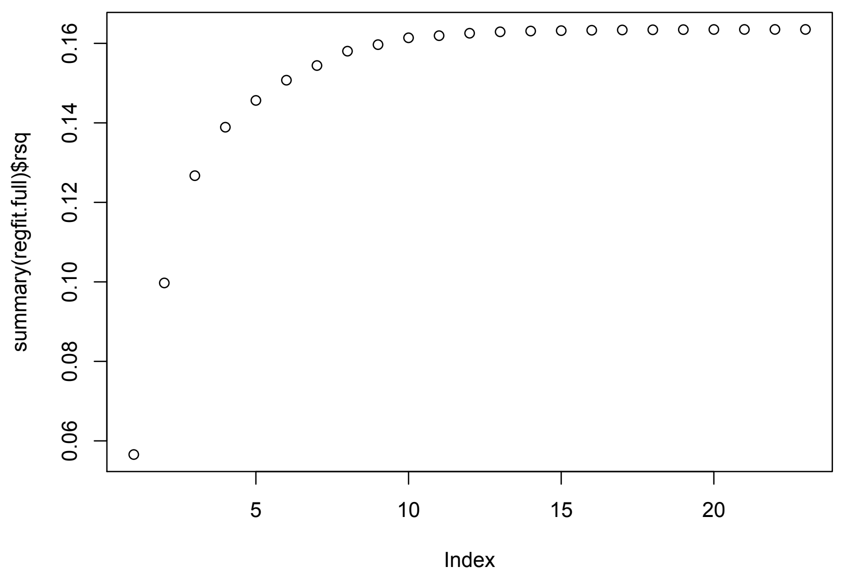
We are going to best subset selection with R.
The package "leaps"

```
# model selection
```

```
data=na.omit(data)

library(leaps)

regfit.full=regsubsets(y~m_edu+f_edu+f_occ+m_occ+birth_area2,data=dat
 a, nvmax=NULL)

summary(regfit.full)
```

We get a matrix showing which variables are chosen for each number of variables. The object obtained has a number of post estimation diagnostic tools. We can plot R^2 against number of variables getting the well known monotonic trend.

```
plot(summary(regfit.full)$rsq)
```



We want to cross validates all 23 models to choose the most appropriate. We do it following the validation set approach (*k*-fold cross validation may be preferable here).

We need a function to predict in the test set all models chosen by the regsubset function. Likely some user (John St. John) wrote a function for us that seems to work perfectly for us:

```
predict.regsubsets = function(object, newdata, id, ...) {
        form  <-  as.formula(~.)
        mat  <-  model.matrix(form, newdata)
        coefi  <-  coef(object, id)
        xvars  <-  names(coefi)
        mat[, xvars] %*% coefi
}
```

Where the code first transforms categorical variables into dummy ("model.matrix") and then extracts the name of the variables used in each model to predict the outcome.

We then select the model with minimum MSE in the test set.

```
# validation

nmds<-length(summary(regfit.full)$rsq)
MSE<-rep(0,nmds)

regfit.full=regsubsets(y~m_edu+f_edu+f_occ+m_occ+birth_area2,data=tra
in, nvmax=NULL)

for (i in 1:nmds) {
     MSE[i]<- sum(test$y - predict.regsubsets(regfit.full, test,i))^2
}

best<-which.min(MSE)

summary(regfit.full)$which[best,][summary(regfit.full)$which[best,]==
TRUE]
```